



C++

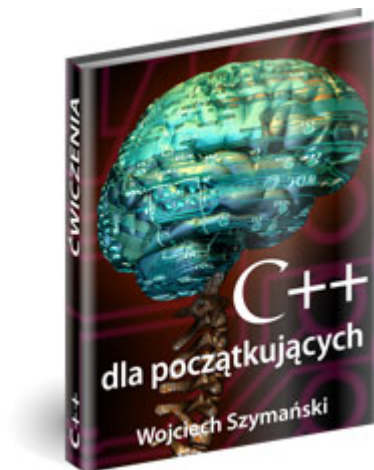
dla początkujących

Wojciech Szymański

C++ DLA POCZĄTKUJĄCYCH

Ćwiczenia

Wojciech Szymański



Wszelkie prawa zastrzeżone. Kopiowanie i rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione.

Autor oraz Wydawnictwo Dobry eBook dołożyli wszelkich starań, aby zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor i Wydawnictwo Dobry eBook nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w niniejszej publikacji.

Wszelkie prawa zastrzeżone © 2007 Dobry eBook

ISBN: 978-83-60863-08-4

Wydanie I

Dobry eBook

ul. Grenadierów 5/5, 30-085 Kraków

tel./fax (12) 353 04 05

e-mail: i.kielar@dobryebook.pl

www.DobryeBook.pl

Spis treści

1	Wstęp	5–7
2	Środowisko pracy	8
	I. Czym jest kompilator	8–9
	II. Wybór programu	9–10
	III. Instalacja programu	10–15
	IV. Korzystanie z programu Dev-C++	15–21
	V. Omówienie najważniejszych funkcji programu	22–24
	VI. Wybrane skróty klawiszowe w Dev-C++	24–25
3	Instrukcje podstawowe	26
	I. Pierwszy program	26–29
	II. Formatowanie tekstu	29–32
	III. Działania na liczbach	32–36
	IV. Typy zmiennych, konwersja i zaokrąglenie	36–38
	V. Kod ASCII	39–40
	VI. Zastosowanie praktyczne: przeliczanie wysokości w stopach na metry	41
	VII. Zastosowanie praktyczne: przeliczanie wieku w latach na dni	42
	VIII. Zastosowanie praktyczne: przeliczanie wieku w latach na dni (dokładniejszy)	43–44
4	Pętle, wyróżnienia warunkowe	45
	I. Instrukcja warunkowa „if”, zakres ważności zmiennych	45–48
	II. Instrukcja warunkowa „while”	49–50
	III. Zastosowanie praktyczne: obliczanie sumy i iloczynu wprowadzonych przez użytkownika liczb	51–52

IV.	Zastosowanie praktyczne: ustalanie najmniejszej i największej liczby spośród wprowadzanych	52–53
V.	Instrukcja warunkowa „do while”	54–55
VI.	Instrukcja warunkowa „for”	56–57
VII.	Zastosowanie praktyczne: obliczanie silni podanej liczby	58–59
VIII.	Zastosowanie praktyczne: wypisanie piramidy z gwiazdek	59–60
IX.	Instrukcja warunkowa „switch”	60–62
X.	Instrukcja „goto”	63–64

5 **Tablice** 65

I.	Wczytywanie elementów do tablicy i ich wypisywanie	65–68
II.	Zastosowanie praktyczne: wypełnianie tablicy liczbami od 0 do 100, z krokiem co 10	68–69
III.	Zastosowanie praktyczne: obliczanie sumy wszystkich elementów tablicy	69–71
IV.	Tablica znakowa. Operator „sizeof”	71–75
V.	Zastosowanie praktyczne: wyszukiwanie najmniejszej liczby w tablicy	75–77
VI.	Kopiowanie zawartości pierwszej tablicy do drugiej (pustej)	77–78
VII.	Kopiowanie zawartości pierwszej tablicy na koniec drugiej	78–79
VIII.	Zastosowanie praktyczne: sprawdzanie czy dwie tablice są takie same	80–82
IX.	Wyszukiwanie ciągu znaków w tablicy	82–84
X.	Zastosowanie praktyczne: sortowanie elementów tablicy	85–86

6 **Funkcje** 87

I.	Ogólne wiadomości na temat funkcji	87–91
II.	Przekazywanie argumentów do funkcji przez wartość	91–93
III.	Zastosowanie praktyczne: wypisywanie kolejnych liczb naturalnych z podanego przedziału	93–94
IV.	Zastosowanie praktyczne: zwracanie wartości większej o 3 od wprowadzonej	94–95
V.	Instrukcja „switch” w funkcji	96–98
VI.	Przeładowanie nazwy funkcji	99–101
VII.	Funkcja z argumentami domniemanymi	101–104
VIII.	Funkcja „inline”	104–106

IX.	Rekurencja: funkcja Fibonacciego	106–108
X.	Rekurencja: mnożenie	108–109
XI.	Rekurencja: potęga	109–110
XII.	Rekurencja: silnia	111–112
7	Wskaźniki	113
I.	Ogólne wiadomości o wskaźnikach	113–115
II.	Zalety i wady wskaźników	115–117
III.	Zastosowanie praktyczne: wypełnianie tablicy liczbami parzystymi z przedziału od 0 do 28	118–119
IV.	Zastosowanie praktyczne: podnoszenie liczby do kwadratu	120–121
V.	Zastosowanie praktyczne: wypisywanie podanego znaku podaną ilość razy	121
VI.	Przekazywanie wyniku funkcji przez wskaźnik	122–123
VII.	Zastosowanie praktyczne: kopiowanie zawartości pierwszej tablicy do drugiej (pustej)	124–126
VIII.	Zastosowanie praktyczne: kopiowanie „n” znaków pierwszej tablicy do drugiej (pustej)	126–127
IX.	Zastosowanie praktyczne: kopiowanie zawartości pierwszej tablicy na koniec drugiej	127–128
X.	Zastosowanie praktyczne: kopiowanie „n” znaków pierwszej tablicy na koniec drugiej	129–130
8	Programowanie obiektowe	131
I.	Klasy i obiekty	131–135
II.	Dane prywatne i publiczne klasy	135–139
III.	Metody klas – wprowadzenie	139–142
IV.	Konstruktory i destruktory – wprowadzenie	142–144
V.	Konstruktor domniemany	145–146
VI.	Konstruktor kopiujący	147–148
VII.	Zastosowanie praktyczne: konstruktory	149–151
9	Skorowidz	152–153

1 Wstęp

Język C++

Język C++, któremu poświęcony jest ten eBook, został stworzony przez Bjarne Stroustrupa w latach osiemdziesiątych XX wieku. Wpływ na jego rozwój i dzisiejszą składnię miały, poza językiem C, przede wszystkim Simula, z której zaczerpnął właściwości obiektowe, a także Algol, Ada, ML i Clu. Początkowo najważniejszą rzeczą wprowadzoną w C++, w stosunku do C, było programowanie obiektowe, później jednak dodawano do niego wiele innych ulepszeń, czyniąc go wygodnym i bardzo elastycznym.

Obecnie w wielu dziedzinach nauki i techniki C++ jest najpopularniejszym językiem programowania, a powiększające się rzesze jego miłośników najlepiej o nim świadczą.

Do podstawowych zalet języka C++ należą:

- nowoczesność,
- duże możliwości,
- użyteczność,
- uniwersalność,
- wspólne cechy z innymi językami programowania,
- elastyczność,
- łatwość nauki.

Dla kogo jest ta książka?

Książkę tą napisałem z myślą o początkujących programistach szukających zbioru praktycznych rozwiązań i zastosowań języka C++. Dzięki szczegółowej analizie pojawiających się – w umiarkowanym tempie – nowych instrukcji, każdy jest w stanie zrozumieć, a co więcej umieć zastosować C++. W publikacji nie ma przykładów instrukcji nie stosowanych, albo bardzo rzadko używanych i w wątpliwych potrzebach.

Nie ma tu także miejsca na omówienia tych elementów języka, które są wykorzystywane przez profesjonalnych programistów. Chciałem, aby ten skrypt był przydatny i zawsze służył pomocą, a przy tym nie składał się z kilkuset stron.

Ponieważ na rynku jest ogromna ilość publikacji dotyczących głównie teorii języka C++, postanowiłem wyjść naprzeciw tym, którzy cenią rozwiązania praktyczne. Stąd wziął się pomysł stworzenia zbioru przykładów.

Jak czytać ten podręcznik?

Od dawna wiadomo, że najlepiej i najszybciej uczymy się poprzez praktykę. Pracując nad tą książką stosowałem się właśnie do tej zasady. Omówienie każdego przykładu składa się z 3 części:

- Kod (źródłowy) programu,
- Na ekranie pojawi się,
- Opis.

Polecam zapoznanie się z każdą z nich. Oczywiście wybór sposobu korzystania z tej książki pozostawiam czytelnikowi i jego indywidualnym potrzebom, ale z własnego doświadczenia wiem, że najlepiej zrozumieć działanie czegoś obserwując wyniki pracy poszczególnych części, czyli w tym wypadku tego, co jest wypisywane na ekranie.

Przykłady zostały ułożone według stopnia trudności i zalecam analizowanie ich po kolei, choć bardziej wtajemniczeni mogą się zadowolić wyrywkową analizą poszczególnych przypadków. Każdy program został opisany językiem prostym i zrozumiałym dla każdego. Celowo unikałem wszelkich trudnych sformułowań, aby nie przysparzać czytelnikowi dodatkowych kłopotów.

Podczas analizy dalszych przykładów zachęcam do próby samodzielnego stworzenia programu i porównania go z tym, który został umieszczony na tych stronach. Pamiętaj jednak, że każde zadanie można rozwiązać na wiele sposobów i każdy z nich może być prawidłowy.

Natraficie też na część pod nazwą „Wskazówka”. W niej zostały przedstawione albo rzeczy ciekawe, albo bardzo ważne, o których zapominać nie wolno.

W publikacji często spotkasz się z określeniem „użytkownik” – tak będę nazywał osobę korzystającą z programu.

W części „Na ekranie pojawi się” to, co zostało wpisane przez użytkownika, jest podkreślone.

Do wykonania programów użyłem kompilatora Bloodshed Dev-C++.

Czytelniku!

Dziękuję Ci za wybranie właśnie mojego poradnika spośród tak bogatego repertuaru książek dotyczących programowania w języku C++. Mam nadzieję, że będziesz z tego wyboru zadowolony i z chęcią wracał będziesz do omawianych na tych stronach przykładów. Wierzę, że za każdym razem, kiedy po tę lekturę sięgniesz, zdasz sobie sprawę, o ile Twoja wiedza się poszerzyła w aspekcie praktycznym.

Ten skrypt dostarczy Ci wiedzy i pomoże zrozumieć prawa rządzące C++, ale pomimo że język ten nie zalicza się do najtrudniejszych, to jednak odrobina wysiłku jest niezbędna do jego zrozumienia i nauczania się.

Przed rozpoczęciem pracy powinienes wiedzieć, że to nie ilość kodu decyduje o użyteczności czy prawidłowości programu. Skromny i prosto napisany kod może okazać się lepszym rozwiązaniem niż skomplikowany i niejasny nawet dla samego twórcy.

2 Środowisko pracy

I. Czym jest kompilator

Mówiąc kompilator, większość z nas ma na myśli gotowe środowisko, w którym tworzymy programy. Jest to mylna definicja, dlatego stwierdziłem, że warto nieco przybliżyć strukturę narzędzia, na którym będziemy pracować.

Środowisko programowania, w którym będziemy tworzyli nasze programy, składa się z trzech różnych mechanizmów, a są nimi:

- Linker,
- Kompilator,
- Edytor.

Po uruchomieniu pewnego środowiska programowania, którym w naszym przypadku będzie Dev-C++, ujrzymy edytor kodu źródłowego. Współczesne edytory udostępniają wiele opcji, dzięki którym tworzenie programu staje się łatwe i przyjemne. Popelniane błędy składniowe są albo samoczynnie korygowane, bądź jest zwracana uwaga programiście. Dodatkową zaletą wielu edytorów kodu źródłowego jest podpowiadanie użytkownikom, co należy wpisać w danym miejscu, a nawet autouzupełnianie kodu.

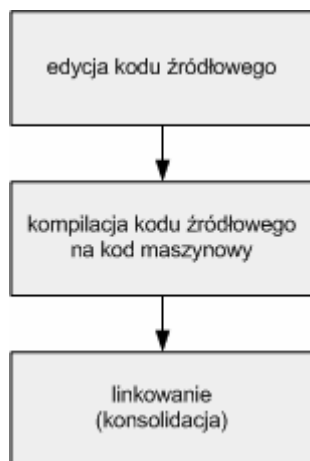
Po stworzeniu kodu źródłowego programu musi on przejść proces kompilacji – czyli zamiany na język zrozumiały dla procesora. To zadanie spoczywa na kompilatorze – dzięki niemu programista nie musi znać języka procesora, czyli kodu maszynowego. Również sam proces pisania programu przebiega szybciej, gdyż zamiast kilku linii kodu maszynowego wprowadzamy jedną linię kodu zrozumiałego dla człowieka. Istnienie narzędzia jakim jest kompilator pozwala na większą przenośność kodu pomiędzy platformami.

Ponieważ często do własnego programu dołącza się różne biblioteki, zawierające dodatkowe instrukcje, lub rozbija się program na kilka plików, niezbędne jest ich dołączenie. Tym zajmuje się linker. Krótko mówiąc łączy on wszystkie części programu w jedną całość – dzięki temu późniejsze korzystanie z programu staje się łatwiejsze i przyjemniejsze.

I tak oto – odpowiadając na postawione w nagłówku pytanie – powstał cały rozdział. Na koniec poprawna definicja kompilatora, która mówi, że jest on składową środowiska

programowania, zajmującą się tylko zamianą kodu źródłowego, napisanego przez człowieka, na kod maszynowy zrozumiały dla procesora.

Poniższy rysunek prezentuje podstawowe etapy tworzenia programu:



Rysunek 2.1 Etapy tworzenia programu

II. Wybór programu

Programów, albo – inaczej mówiąc – środowisk, w których tworzy się programy, jest bardzo dużo. Mamy do wyboru rozwiązania komercyjne i darmowe, proste i trudne w obsłudze, ładne i brzydkie... Preferencje każdego programisty zmieniają się z upływem czasu i z konieczności korzystania z pewnych konkretnych funkcji.

Pierwszym i zarazem najważniejszym kryterium wyboru, przed jakim staje programista, jest uzależnienie od platformy systemowej. Programu pracującego pod Windows nie da się uruchomić na Linuksie.

Zacznę od krótkiego omówienia oprogramowania pracującego na platformie linuksowej. Najbardziej popularnym kompilatorem C++ używanym w systemie Linux jest g++ wydawany przez projekt GNU. Jest to tylko kompilator, niepołączony z żadnym edytorem, co dla początkujących programistów jest uciążliwe. Jesteśmy zmuszeni tworzyć kody źródłowe w edytorze tekstowym, następnie zlecać g++ przeprowadzenie kompilacji. Kompilator g++ jest standardowo dołączany do dystrybucji Linuksa.